(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 August 2005 (25.08.2005)

PCT

(10) International Publication Number
WO 2005/078575 A2

(51) International Patent Classification[7]: G06F 9/38

(21) International Application Number:
PCT/US2005/001281

(22) International Filing Date: 14 January 2005 (14.01.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/772,750 4 February 2004 (04.02.2004) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **YAMADA, Koichi** [JP/US]; 982 Cherrystone Drive, Los Gatos, CA 95032 (US). **KAY, Allen** [US/US]; 2780 Mauricia Avenue, Santa Clara, CA 95051 (US).

(74) Agent: **VINCENT, Lester, J.**; Blakely, Sokoloff, Taylor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SHARING IDLED PROCESSOR EXECUTION RESOURCES

(57) Abstract: A processor including a plurality of logical processors, and an instruction set, the instruction set including of one or more instructions which when executed by a first logical processor, cause the first logical processor to make a processor execution resource previously reserved for the first processor available to a second processor in the plurality of processors in response to the first logical processor being scheduled to enter an idle state.

WO 2005/078575 A2

Sharing Idled Processor Execution Resources

Background

[01] In the high level view of a processor depicted in Fig. 1a, a processor may be conceptualized as being comprised of two components, the first implementing the architectural state of the processor, such as for example its registers and program counter, and the second composed of processor execution resources, such as, for example, a translation lookaside buffer (TLB).

[02] In one type of multiprocessing processor based system, as depicted in Fig. 1b, multiple physical processors are interconnected by a bus system, and each physical processor maintains a separate architectural state in hardware as well as a separate set of processor execution resources in hardware. In a thread scheduling scenario where each processor of such a system is scheduled to execute a different thread, an instance may arise when one of the processors in the system is idled because it is waiting on a slower device in the system, such as a disk drive, or because it is currently not scheduled to execute a thread. In this instance, the processor and all of its execution resources are also idled and unavailable to other processors of the system.

[03] In another type of processor based system such as that depicted in Fig. 1c, a hardware processor that maintains separate architectural states in the processor's hardware for a plurality of logical processors may, however, have a single processor core pipeline that is shared by the logical processors and a single set of processor execution resources, including the TLB, that is shared by the logical processors. Such a processor architecture is exemplified by the Intel® Xeon™ processor with Hyper Threading Technology, among others, and is well known in the art.

[04] In such a logical multiprocessing system, a thread scheduler may schedule a different thread to execute on each of the logical processors because each logical processor maintains its architectural state separately from all other logical processors. When a logical processor is idled by an operating system thread scheduler or is waiting

for data from a slow storage device, it may either execute an idle task, typically a tight loop, and periodically check for an interrupt; or it may suspend its activity and wait for a wake up signal of some type to resume execution of a thread.

[05] In contrast to a multiprocessing system where processor execution resources are physically separated, in this type of logical multiprocessing system, when one of the multiple logical processors in such a system is idled, dynamically allocated processor execution resources that are not being used by the idled logical processor may be available to other logical processors that are currently executing threads for the user or the system.

[06] Processor execution resources in a logical multiprocessing system may, however, be reserved for a logical processor. This may occur in different ways. For one example, a logical processor may lock a dynamically allocated processor execution resource such as a translation register (TR) from the TLB thus making it unavailable to other logical processors. In another instance, the logical processor may be statically allocated processor execution resources such as TCs and thus these statically allocated resources may be unavailable to other logical processors. These reserved resources typically continue to be unavailable to other logical processors even after the logical processor for which they are reserved is idled. Thus, TRs that are locked by a logical processor generally continue to be locked by the logical processor while it is idling; and statically allocated TCs allocated to the logical processor continue to be statically allocated to the logical processor while it is idling.

Brief Description of the Drawings

Figure 1 Depicts high level views of different types of processor architectures.

Figure 2 is a flowchart of processing in one embodiment.

Figure 3 depicts a processor based system in one embodiment.

Detailed Description

[07] In one embodiment processing occurs as depicted in the high level flowchart in Fig. 2. In the figure, two logical processors, Processor 1, 200, and Processor 2, 205, are executing threads scheduled by an operating system that includes a thread scheduler 210. At 215, Processor 1 is switched out from an executing thread due to, for instance, termination of the thread or a page fault, and returns to the thread scheduler. If no more tasks are scheduled for this logical processor, 220, the processor executes an idling sequence, 225-230. First, the logical processor gives up any reserved processor execution resources held by the logical processor 225, releasing them to the common pool 260. Thus for example, Processor 1 may return a Translation Cache entry or Translation Cache Register to the general pool of registers in the Translation Lookaside Buffer.

[08] In different embodiments, the processing in step 225 may differ. In some embodiments, the exclusively held resource released may be a dynamically allocated resource and have previously been locked by Processor 1. In such an embodiment, in step 225, the logical processor unlocks the resource and thereby makes it available to other logical processors. In another embodiment, the exclusively held resource may have been previously statically allocated to Processor 1. In such embodiments, in step 225, the statically allocated resource is deallocated and is returned to the pool of dynamically allocated resources 260.

[09] After Processor 1 enters an idled state, such as a state of suspension 230 in this embodiment, it may be requested for execution of a new or resumed thread by a wake up signal such as an interrupt 235. In other embodiments the processor may enter an idle task loop instead of the suspension depicted at 230 and periodically check for interrupts.

[10] Following the wake up signal, the logical processor then re-acquires the exclusively reserved resources by either locking or statically allocating them to itself

as necessary, 240. The logical processor then switches to an incoming thread and continues execution of that thread, 245.

[11] The resources freed by Processor 1 before suspension or idling at 225 become available to another logical processor such as Processor 2, 205, executing a thread such as the depicted user thread 250. These resources may then be dynamically allocated to the logical processor as necessary from the pool of shared processor execution resources during the execution of the thread, 255.

[12] Fig. 3 depicts a processor based system in one embodiment where the logical processors are implemented as part of a processor 300. Programs that execute on the logical processors are stored in memory 340 connectively coupled to the processor by bus system 320. The memory may include a non-volatile memory section storing firmware that includes a thread scheduler performing processing substantially as described above.

[13] Many other embodiments are possible. For instance, while the above description limits itself to logical processors, similar processing is applicable to physically separate multiprocessors that share any common execution resources. In such embodiments, a hybrid version of logical and physical multiprocessing is implemented where separate architectural states and some execution resources are separated in hardware, but other execution resources are shared in hardware and may be released using processing similar to that depicted in Fig. 2. In some embodiments, the thread scheduler referenced above may form a component of firmware resident in non-volatile memory as depicted in Fig. 3, while in others it may be a portion of operating system software stored on disk media accessible to the processor. In some embodiments, the actions taken to release and reserve processor execution resources may be directly implemented in hardware and ancillary to the processor's instruction execution system, while in other embodiments they may be actions taken by the processor as part of the execution of one or more instructions. In some embodiments

the shared execution resources may include special purpose registers unrelated to the TLB. Embodiments are not limited to two processors, three or more processors may share execution resources and perform processing analogous to the processing described above.

[14] Embodiments in accordance with the claimed subject matter may be provided as a computer program product that may include a machine-readable medium having stored thereon data which when accessed by a machine may cause the machine to perform a process according to the claimed subject matter. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, DVD-ROM disks, DVD-RAM disks, DVD-RW disks, DVD+RW disks, CD-R disks, CD-RW disks, CD-ROM disks, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnet or optical cards, flash memory, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, embodiments may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer to a requesting computer by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[15] Many of the methods are described in their most basic form but steps can be added to or deleted from any of the methods and information can be added or subtracted from any of the described messages without departing from the basic scope of the claimed subject matter. It will be apparent to those skilled in the art that many further modifications and adaptations can be made. The particular embodiments are not provided to limit the invention but to illustrate it. The scope of the claimed subject matter is not to be determined by the specific examples provided above but only by the claims below.

Claims

What is claimed is:

1. A method comprising:

in a processor based system where a plurality of processors share processor execution resources, in response to a first processor in the plurality of processors being scheduled to enter an idle state, making a processor execution resource previously reserved for the first processor available to a second processor in the plurality of processors.

2. The method of claim 1 further comprising reserving the processor execution resource for the first processor in response to the first processor being scheduled to execute a task.

3. The method of claim 2 wherein each of the plurality of processors is a logical processor of the processor based system.

4. The method of claim 3 wherein the first processor being scheduled to enter an idle state further comprises the first processor executing a processor instruction requesting the first processor to enter an idle state.

5. The method of claim 4 wherein making the processor execution resource previously reserved for the first processor available to a second processor further comprises releasing the processor execution resource into a common pool of processor execution resources accessible from the second processor.

6. The method of claim 5 wherein the first processor being scheduled to execute a task further comprises the first processor receiving a wake up signal.

7. The method of claim 6 wherein the processor execution resource previously reserved for the first processor further comprises the processor execution resource previously statically allocated to the first processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises de-allocating the processor execution resource.

8. The method of claim 6 wherein the processor execution resource previously reserved for the first processor further comprises the processor execution resource previously locked by the first processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises the first processor unlocking the processor execution resource.

9. The method of claim 6 wherein the common pool of processor execution resources comprises a translation lookaside buffer and the processor execution resource is a translation cache entry from the translation lookaside buffer.

10. A processor comprising:

a plurality of logical processors; and

an instruction set, the instruction set comprising one or more instructions which when executed by a first logical processor, cause the first logical processor to make a processor execution resource previously reserved for the first processor available to a second processor in the plurality of processors in response to the first logical processor being scheduled to enter an idle state.

11. The processor of claim 10 wherein to the first logical processor being scheduled to enter an idle state further comprises the first processor executing a processor instruction requesting the first logical processor to enter an idle state.

12. The processor of claim 11 wherein causing the first logical processor to make the processor execution resource previously reserved for the first logical processor available to a second logical processor further comprises releasing the processor execution resource into a common pool of processor execution resources accessible from the second logical processor.

13. The processor of claim 12 wherein the processor execution resource previously reserved for the first logical processor further comprises the processor execution resource previously statically allocated to the first logical processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises de-allocating the processor execution resource.

14. The processor of claim 12 wherein the processor execution resource previously reserved for the first logical processor further comprises the processor execution resource previously statically allocated to the first logical processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises the first processor unlocking the processor execution resource.

15. A system comprising:

a processor, the processor comprising

a plurality of logical processors; and

an instruction set, the instruction set comprising one or more instructions which when executed by a first logical processor, cause the first logical processor to make a processor execution resource previously reserved for the first processor available to a second processor in the plurality of processors in response to the first logical processor being scheduled to enter an idle state;

firmware to schedule the first logical processor to enter an idle state; and

a bus to interconnect the firmware and the processor.

16. The system of claim 15 wherein the first logical processor being scheduled to enter an idle state further comprises the first processor executing a processor instruction requesting the first logical processor to enter an idle state.

17. The system of claim 16 wherein causing the first logical processor to make the processor execution resource previously reserved for the first logical processor available to a second logical processor further comprises releasing the processor execution resource into a common pool of processor execution resources accessible from the second logical processor.

18. The system of claim 17 wherein the processor execution resource previously reserved for the first logical processor further comprises the processor execution resource previously statically allocated to the first logical processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises de-allocating the processor execution resource

19. The system of claim 17 wherein the processor execution resource previously reserved for the first logical processor further comprises the processor execution resource previously statically allocated to the first logical processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises the first processor unlocking the processor execution resource.

20. A machine accessible medium having stored thereon data which when accessed by a machine causes the machine to perform a method, the method comprising:
in a processor based system where a plurality of processors share processor execution resources, in response to a first processor in the plurality of processors being scheduled to enter an idle state, making a processor execution resource previously reserved for the first processor available to a second processor in the plurality of processors.

21. The machine accessible medium of claim 20 further comprising reserving the processor execution resource for the first processor in response to the first processor being scheduled to execute a task.

22. The machine accessible medium of claim 21 wherein each of the plurality of processors is a logical processor of the processor based system.

23. The machine accessible medium of claim 22 wherein the first processor being scheduled to enter an idle state further comprises the first processor executing a processor instruction requesting the first processor to enter an idle state.

24. The machine accessible medium of claim 23 wherein making the processor execution resource previously reserved for the first processor available to a second processor

further comprises releasing the processor execution resource into a common pool of processor execution resources accessible from the second processor.

25. The machine accessible medium of claim 24 wherein the first processor being scheduled to execute a task further comprises the first processor receiving a wake up signal.

26. The machine accessible medium of claim 25 wherein the processor execution resource previously reserved for the first processor further comprises the processor execution resource previously statically allocated to the first processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises de-allocating the processor execution resource.

27. The machine accessible medium of claim 25 wherein the processor execution resource previously reserved for the first processor further comprises the processor execution resource previously locked by the first processor; and wherein releasing the processor execution resource into a common pool of processor execution resources further comprises the first processor unlocking the processor execution resource.

28. The machine accessible medium of claim 25 wherein the common pool of processor execution resources comprises a translation lookaside buffer and the processor execution resource is a translation cache entry from the translation lookaside buffer.

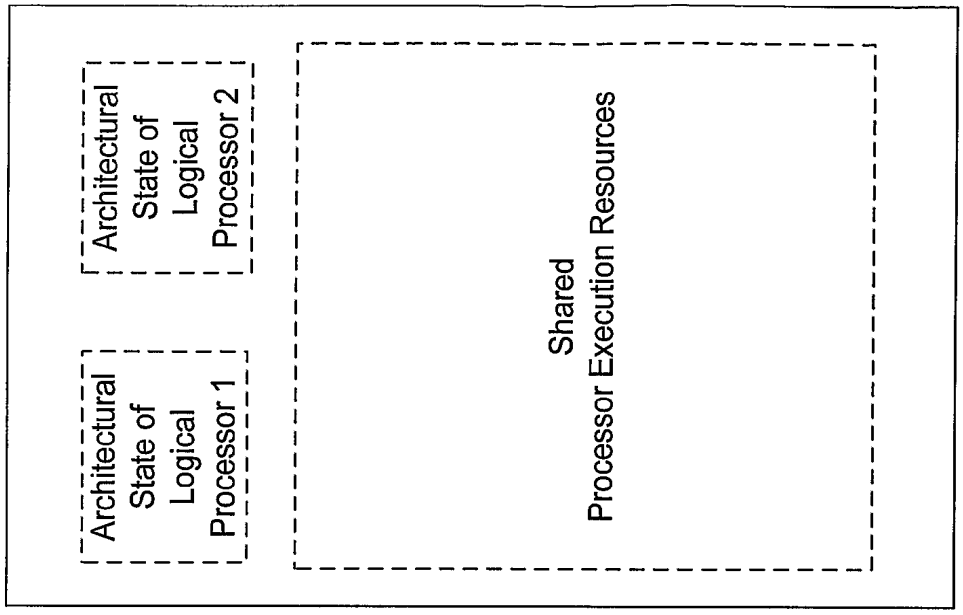

Figure 1c

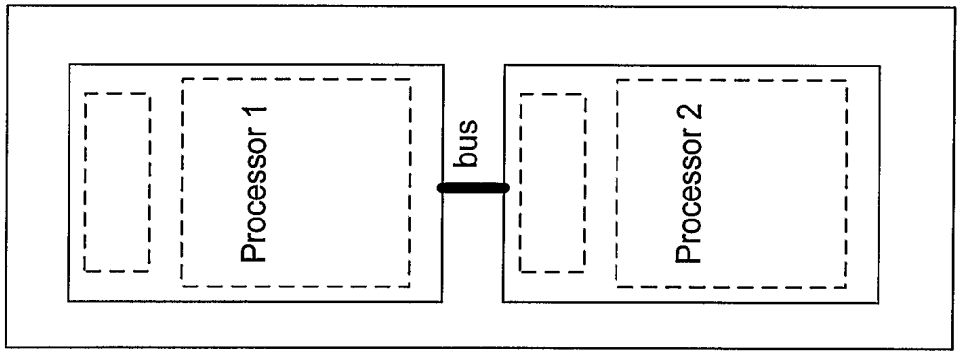

Figure 1b

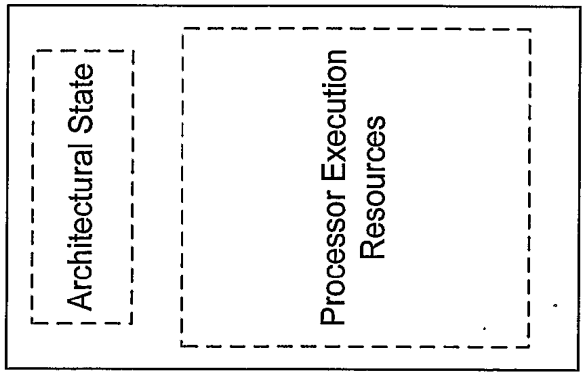

Figure 1a

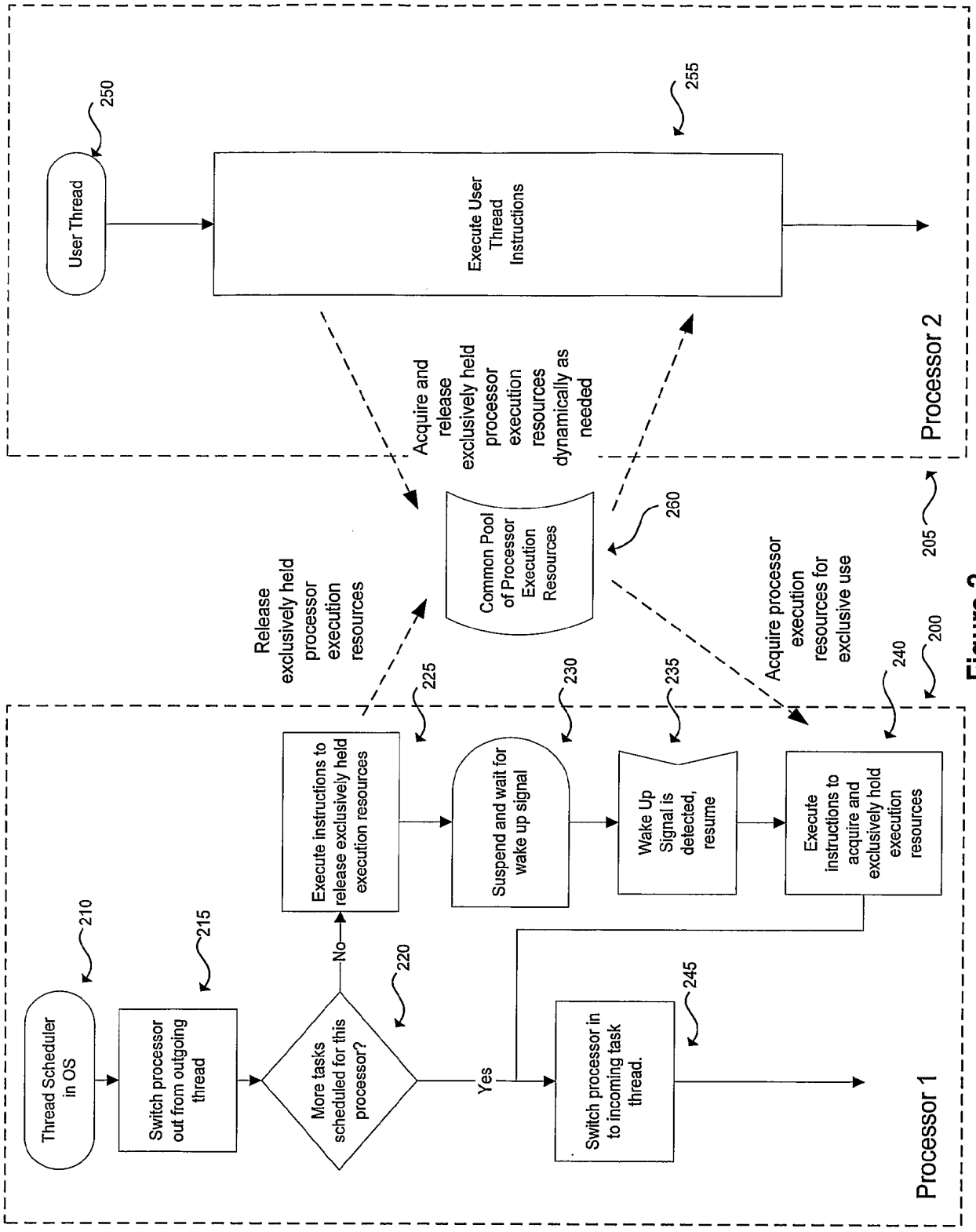
Processor 1
Thread Scheduler in OS
210
Switch processor out from outgoing thread
215
More tasks scheduled for this processor?
220
No
Yes
Execute instructions to release exclusively held execution resources
225
Suspend and wait for wake up signal
230
Wake Up Signal is detected, resume
235
Execute instructions to acquire and exclusively hold execution resources
240
Switch processor in to incoming task thread.
245
Release exclusively held processor execution resources
Acquire processor execution resources for exclusive use
Common Pool of Processor Execution Resources
260
205
200
Processor 2
User Thread
250
Execute User Thread Instructions
255
Acquire and release exclusively held processor execution resources dynamically as needed

Figure 2

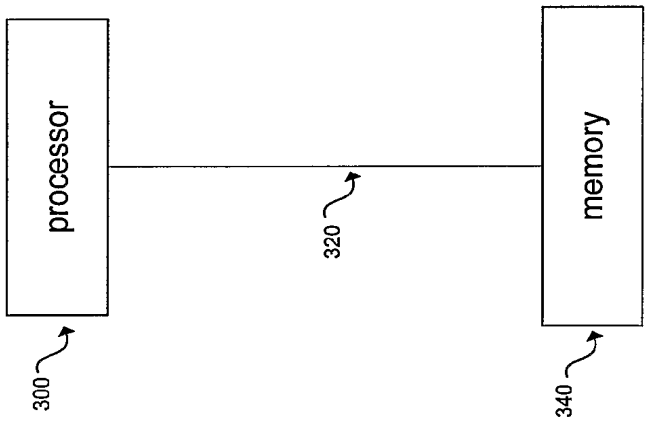

Figure 3